



COMP90042

Web search and text analysis

Workshop Week 9



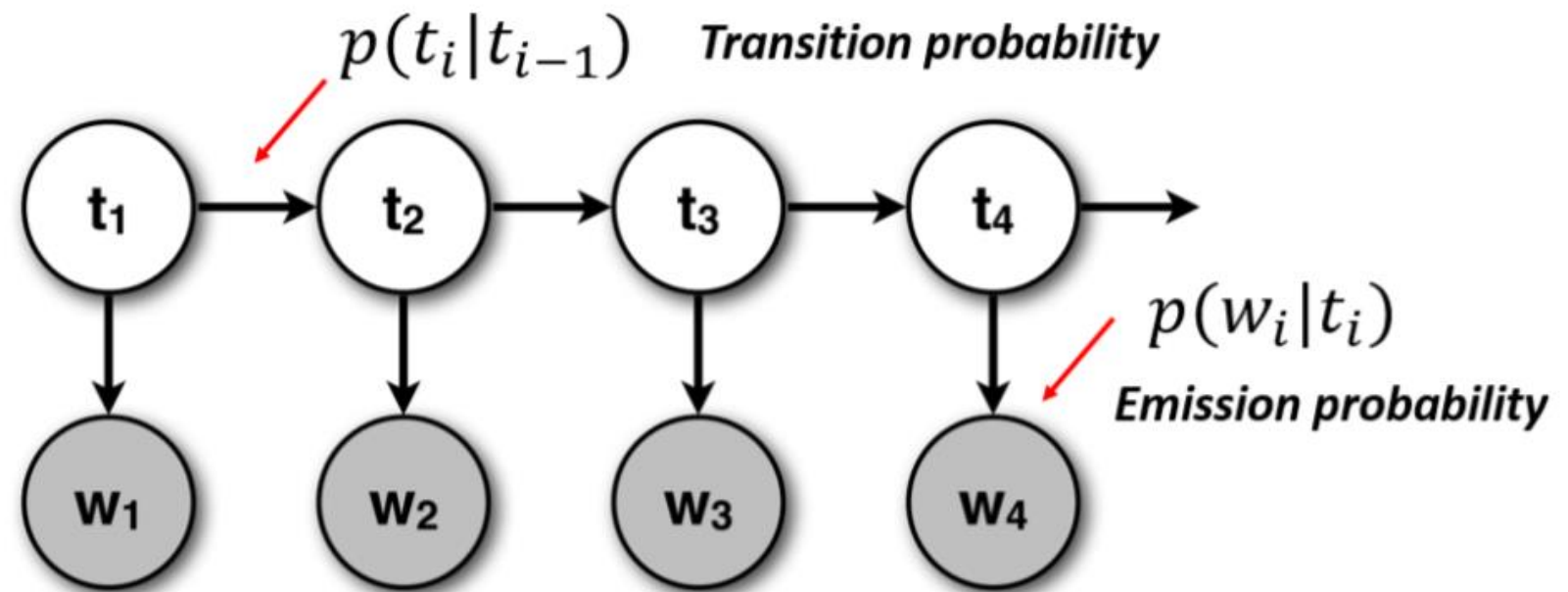
Your tutor

- Winn Chow (Senior Tutor)
- winn.chow1@unimelb.edu.au
- Office: Doug McDonell - 9.23
- Here, you can find my workshop slides:
- <https://github.com/winnchow/COMP90042-Workshops>

Q1

1. What are the assumptions that go into a **Hidden Markov Model**? What is the time complexity of the **Viterbi algorithm**? Is this practical?
 - (a) How can an HMM be used for POS tagging a text? For the purposes of POS tagging:
 - i. How can the initial state probabilities π be estimated?
 - ii. How can the transition probabilities A be estimated?
 - iii. How can the emission probabilities B be estimated?
 - (b) Estimate π , A and B for POS tagging, based on the following corpus:
 1. silver-JJ wheels-NNS turn-VBP
 2. wheels-NNS turn-VBP right-JJ
 3. right-JJ wheels-NNS turn-VBP

HMM



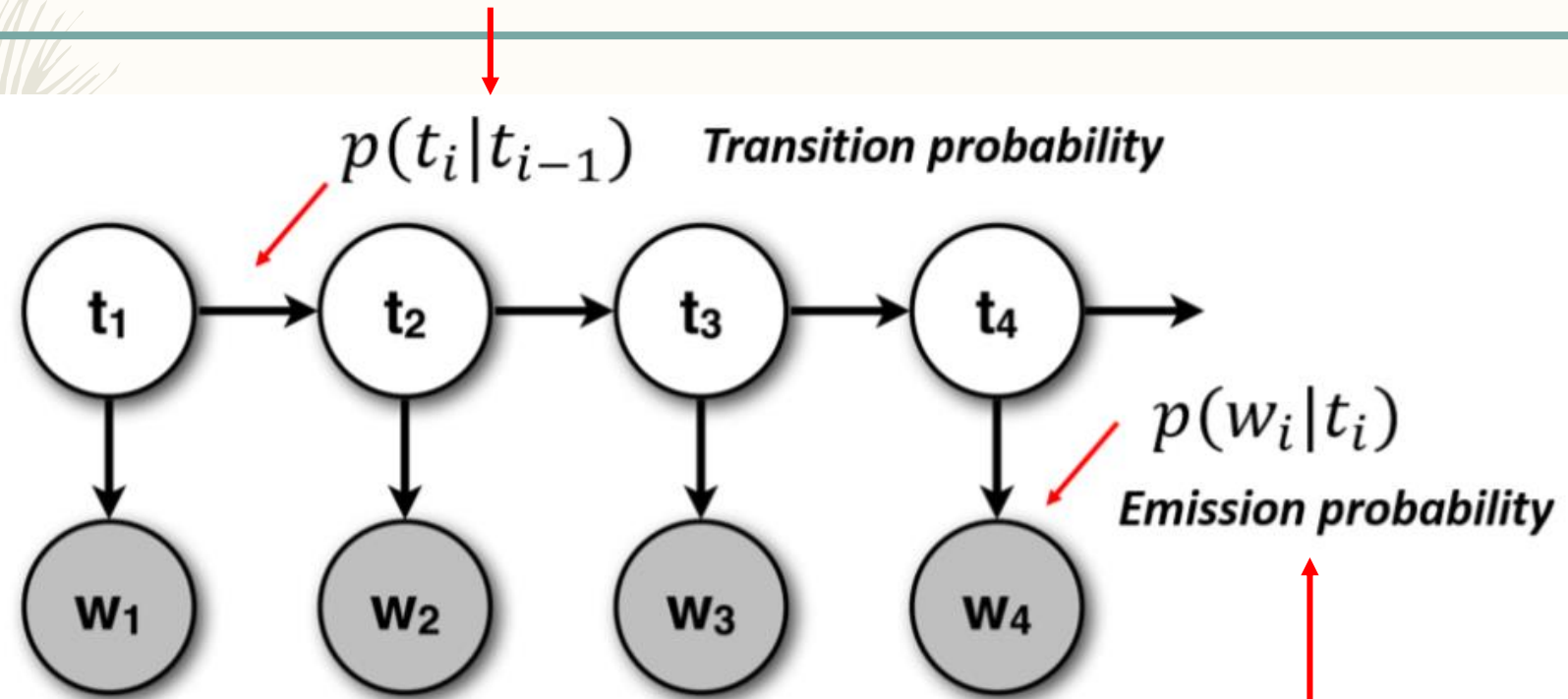
HMM

- **Markov assumption:** the likelihood of transitioning into a given state depends only on the current state, and not the previous state(s) (or output(s))

$$O(T^2W)$$

T – # of states

W – sentence length



- **Output independence assumption:** the likelihood of a state producing a certain word (as output) does not depend on the preceding (or following) state(s) (or output(s)).



Q1b

1. silver-JJ wheels-NNS turn-VBP
 2. wheels-NNS turn-VBP right-JJ
 3. right-JJ wheels-NNS turn-VBP
-

– Initial state probabilities: $\pi[\text{JJ}; \text{NNS}; \text{VBP}]$

$$\pi[\text{JJ}, \text{NNS}, \text{VBP}] = [\frac{2}{3}, \frac{1}{3}, 0]$$



Q1b

1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP

– Transition probabilities: **A**

	JJ	NNS	VBP
JJ			
NNS			
VBP			

Q1b

1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP

– Transition probabilities: **A**

	JJ	NNS	VBP
JJ	0	1	0
NNS	0	0	1
VBP	1	0	0

Q1b

1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP

– Emission probabilities: **B**

	right	Silver	turn	wheels
JJ				
NNS				
VBP				

Q1b

1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP

– Emission probabilities: **B**

	right	Silver	turn	wheels
JJ	2/3	1/3	0	0
NNS	0	0	0	1
VBP	0	0	1	0

Q2

2. Consider using the following Hidden Markov Model to tag the sentence `silver wheels turn`:

$$\pi[\text{JJ}, \text{NNS}, \text{VBP}] = [0.3, 0.4, 0.3]$$

<i>A</i>	JJ	NNS	VBP	<i>B</i>	silver	wheels	turn
JJ	0.4	0.5	0.1	JJ	0.8	0.1	0.1
NNS	0.1	0.4	0.5	NNS	0.3	0.4	0.3
VBP	0.4	0.5	0.1	VBP	0.1	0.3	0.6

(a) Visualise the HMM as a graph.

(b) Use the **Viterbi algorithm** to find the most likely tag for this sequence.

Q2

	silver	wheels	turn
JJ	$\pi[\text{JJ}] * B[\text{JJ}, \text{silver}]$ = P1	P1 * A[JJ, JJ] * B[JJ, wheels] = P1_1 P2 * A[NNS, JJ] * B[JJ, wheels] = P1_2 P3 * A[VBP, JJ] * B[JJ, wheels] = P1_3 (Pick the largest)	
NNS	P2		
VBP	P3		


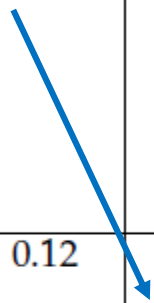
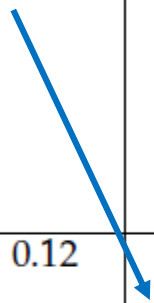
Q2

α		1:silver	2:wheels	3:turn
JJ:	JJ	$\pi[\text{JJ}]B[\text{JJ}, \text{silver}]$ $0.3 \times 0.8 = 0.24$		
NNS:	NNS	$\pi[\text{NNS}]B[\text{NNS}, \text{silver}]$ $0.4 \times 0.3 = 0.12$		
VBP:	VBP	$\pi[\text{VBP}]B[\text{VBP}, \text{silver}]$ $0.3 \times 0.1 = 0.03$		

Q2

α	1:silver		2:wheels	3:turn
JJ:	0.24	JJ \rightarrow JJ 0.24	$A[\text{JJ}, \text{JJ}]B[\text{JJ}, \text{wheels}]$ $\times 0.4 \times 0.1 = \mathbf{0.0096}$	
		NNS \rightarrow JJ 0.12	$A[\text{NNS}, \text{JJ}]B[\text{JJ}, \text{wheels}]$ $\times 0.1 \times 0.1 = 0.0012$	
		VBP \rightarrow JJ 0.03	$A[\text{VBP}, \text{JJ}]B[\text{JJ}, \text{wheels}]$ $\times 0.4 \times 0.1 = 0.0012$	
NNS:	0.12	JJ \rightarrow NNS 0.24	$A[\text{JJ}, \text{NNS}]B[\text{NNS}, \text{wheels}]$ $\times 0.5 \times 0.4 = \mathbf{0.048}$	
		NNS \rightarrow NNS 0.12	$A[\text{NNS}, \text{NNS}]B[\text{NNS}, \text{wheels}]$ $\times 0.4 \times 0.4 = 0.0192$	
		VBP \rightarrow NNS 0.03	$A[\text{VBP}, \text{NNS}]B[\text{NNS}, \text{wheels}]$ $\times 0.5 \times 0.4 = 0.006$	
VBP:	0.03	JJ \rightarrow VBP 0.24	$A[\text{JJ}, \text{VBP}]B[\text{VBP}, \text{wheels}]$ $\times 0.1 \times 0.3 = 0.0072$	
		NNS \rightarrow VBP 0.12	$A[\text{NNS}, \text{VBP}]B[\text{VBP}, \text{wheels}]$ $\times 0.5 \times 0.3 = \mathbf{0.018}$	
		VBP \rightarrow VBP 0.03	$A[\text{VBP}, \text{VBP}]B[\text{VBP}, \text{wheels}]$ $\times 0.1 \times 0.3 = 0.0009$	

Q2

α	1:silver	2:wheels		3:turn 
JJ:	0.24 	0.0096	JJ → JJ	$A[\text{JJ}, \text{JJ}] B[\text{JJ}, \text{turn}]$
		JJ → JJ	0.0096	$\times 0.4 \times 0.1 = 0.000384$
			NNS → JJ	$A[\text{NNS}, \text{JJ}] B[\text{JJ}, \text{turn}]$
			0.048	$\times 0.1 \times 0.1 = 0.00048$
NNS:	0.12 		VBP → JJ	$A[\text{VBP}, \text{JJ}] B[\text{JJ}, \text{turn}]$
			0.018	$\times 0.4 \times 0.1 = \mathbf{0.00072}$
		0.048	JJ → NNS	$A[\text{JJ}, \text{NNS}] B[\text{NNS}, \text{turn}]$
		JJ → NNS	0.0096	$\times 0.5 \times 0.3 = 0.00144$
VBP:	0.03		NNS → NNS	$A[\text{NNS}, \text{NNS}] B[\text{NNS}, \text{turn}]$
			0.048	$\times 0.4 \times 0.3 = \mathbf{0.00576}$
			VBP → NNS	$A[\text{VBP}, \text{NNS}] B[\text{NNS}, \text{turn}]$
			0.018	$\times 0.5 \times 0.3 = 0.0027$
		0.018	JJ → VBP	$A[\text{JJ}, \text{VBP}] B[\text{VBP}, \text{turn}]$
		NNS → VBP	0.0096	$\times 0.1 \times 0.6 = 0.000576$
			NNS → VBP	$A[\text{NNS}, \text{VBP}] B[\text{VBP}, \text{turn}]$
			0.048	$\times 0.5 \times 0.6 = \mathbf{0.0144}$
			VBP → VBP	$A[\text{VBP}, \text{VBP}] B[\text{VBP}, \text{turn}]$
			0.018	$\times 0.1 \times 0.6 = 0.00108$

Q3

3. What are regular grammar and regular language? How are they different?
 - (a) Regular languages are closed under union, intersection and concatenation. What does it mean? Why is it important?
 - (b) Draw a Finite State Acceptor (FSA) for word morphology to show the possible derivations from root forms using the words: play, played, playing; walk, walked, walking; sit, sat, sitting.
 - (c) What are Weighted Finite State Acceptors (WFSAs)? When and why are they useful?

Q3

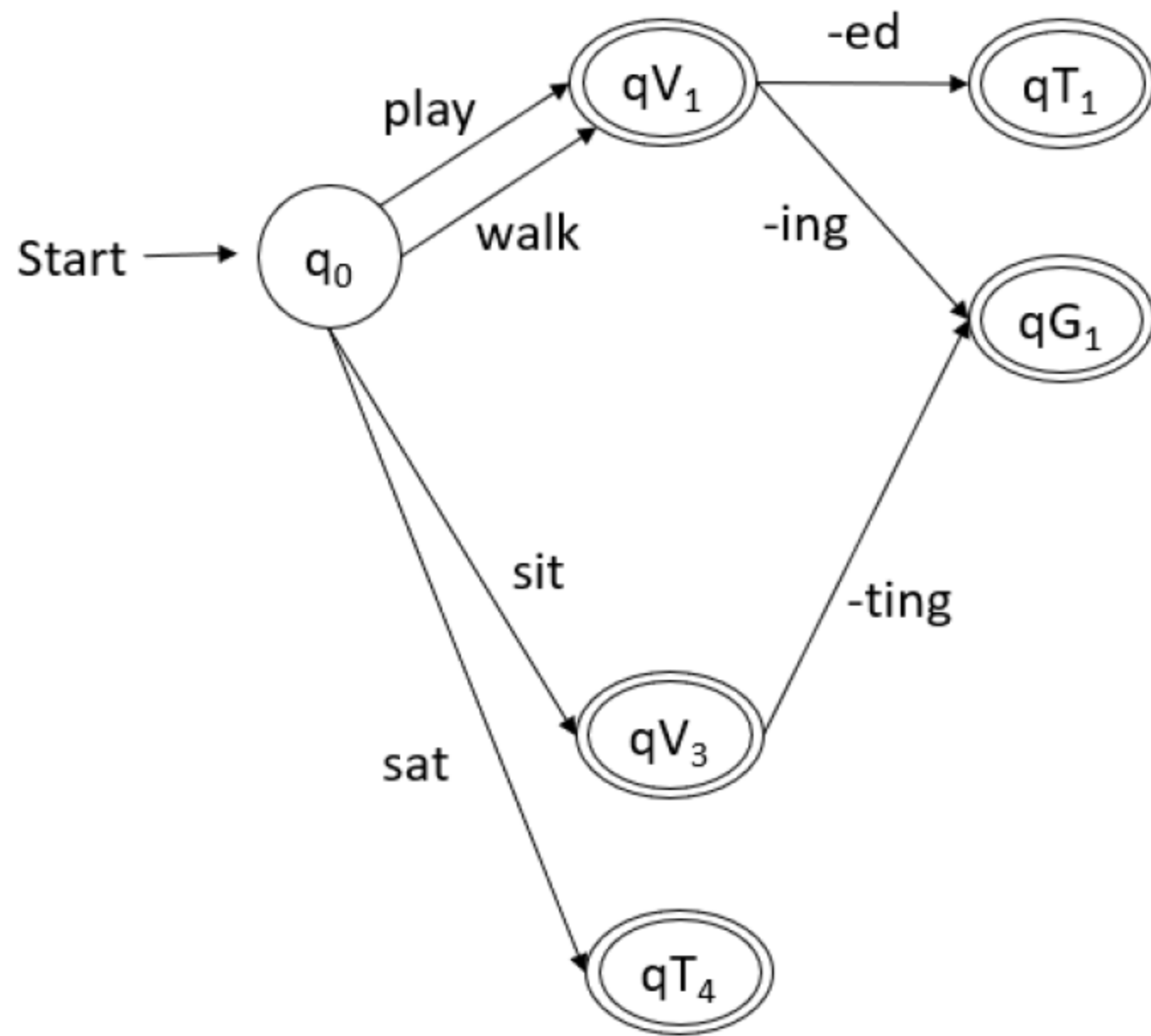
- A language is a set of acceptable strings and a grammar is a generative description of a language.
- Regular language is a formal language that can be expressed using a regular expression.
- Regular grammar is a formal grammar defined by a set of productions rules in the form of $A \rightarrow xB$, $A \rightarrow x$ and $A \rightarrow \epsilon$, where A and B are non-terminals, x is a terminal and ϵ is the empty string.
- A language is regular if and only if it can be generated by a regular grammar.

Q3

- For example: A simple regular grammar
 - Rules: $S \rightarrow A, A \rightarrow a A, A \rightarrow \epsilon$
 - S is the start symbol
 - It will generate words such as a, aa, aaa, aaa .
 - The set of words generated by this regular grammar is a regular language.
 - This regular language can also be expressed in regular expression $(a)^*$.

Q3a

- This means that if $L1$ and $L2$ are two regular languages, then $L1 \cup L2$, $L1 \cap L2$, and the language strings that are the concatenation of $L1$ and $L2$ are also regular languages.
- This closure property allows us to apply operations on regular languages to produce a regular language.
- This allows for NLP problems to be factored into small simple parts, such that we can develop regular languages for each part, and combine them into a complex system to handle the NLP problems. This is particularly relevant for transducers and the composition operation, which are used in many NLP pipelines. (Note that FSTs implement “regular relations” rather than regular languages, but the distinction is a bit subtle and is not something we will dive into.)





Q3c

- WFSAs are generalizations of FSAs, with each path assigned a score, computed from the transitions, the initial state, and the final state. The total score for any path is equal to the sum of the scores of the initial state, the transitions, and the final state.
- WFSAs can produce a score for each valid word for sub-word decomposition problems or sentence for words-in-sentence problems, while FSAs have no way to express preferences among technically valid words or sentences.
- For example, WFSAs can assign scores to all strings of characters forming words, so that spelling mistakes, new words, or strange-but-acceptable words can still be handled.
- The same argument holds for sequences of words forming sentences. Clearly some word sequences are gibberish, but being able to provide a numerical score can help in many applications, like how LMs can be used in sentence generation, speech recognition, OCR, translation, etc.